transcoding (format conversion) in the form of a framework that can be used by the infolet provider. For example, the blog infolet **16** converts a blog entry submitted by a user through any of the gateways **12** into a blog information item **18**. RSS-enabled infolets **14***a, b* and *c,* operating within each server(s) **22**, implement protocol interfaces that access various information sources (such as a location service **24**, RSS source service **25**, sensor service **26**, etc.). Such infolets **14** also include and implement a module that converts the retrieved information from the various services **24** and **26** into a properly formatted RSS data feed. In general, the conversion of retrieved information into a format that facilitates creating RSS feeds is performed in a timely manner by an infolet making the data amenable for aggregation with other sources of information. The blog infolet **16** is also capable of presenting its information in an RSS feed.

[0031] The aggregator infolet **38** assembles RSS feeds from other infolets to provide a unique aggregated RSS feed **28**. This aggregated feed **28** is then ready for querying, filtering or publishing to other media gateways. Note that the aggregator infolet **38** can be any application infolet that uses a third party aggregator library and handles the actual aggregation task. Content from the aggregated feed **28** can be adapted and provided for delivery to subscribers who are interested in any event reported in the aggregated RSS feed **28**. The aggregation itself takes into account user preferences and/or the user's service profile that may be specified as weights of the aggregation criteria. This aggregation procedure is a weighted clustering mechanism. The publication infolet **30**, for example, is used to publish personalized and filtered RSS content from the aggregated feed **28** onto a user's personal or group blog site **32** maintained by the platform **10**. A delivery module inside the blog infolet **16** or the aggregator infolet **38** is used to adapt user selected portions of the aggregated RSS feed (**18** or **28**) for delivery as content to users' mobile or other gateway **12** related devices.

[0032] A user's blog description schema is where a user defines the attributes used to describe the data items. Such attributes may include, but are not limited to a time, location, direction, type, hobby, event, genre, species, culture, religion, size, shape, color, any physical attribute, topic, etc. For example, a user may have a blog site that is about Ford Mustang automobiles. The user may define attributes for the blog item descriptors based on an automobile brand, model, year, color, part, date, geographic location, VIN, level of customization, just to name a few. These user defined attributes may not be related or similar to standardized or syndicated RSS or ATOM data attributes that are commonly accepted by the media houses that produce the standardized or syndicated RSS or ATOM feeds. A 'description schema' can be defined by each user for his or her blogs and contains descriptive information as a set of attributes constrained by type and type specific restrictions. A user's service profile can be defined as where user preferences with respect to a particular service are stored. For example, the default blog to be used when none is specified in a user posting is part of the user's profile with respect to the Blog infolet. A user's service profile can also contain, in this particular case, shortcuts to predefined user queries, etc.

[0033] In FIG. **4**, exemplary blog item attributes are described. Every blog and blog item can be annotated with a set of name-value pairs. The names and values may be established by the creator of the blog or may partially come from the data items in the blog. A name together with an associated value type make up a 'description schema attribute'; a set of such attributes makes up a descriptor. Thus, a blog or blog item is annotated by a set of descriptions, wherein a description is an instance of a descriptor (i.e., an actual value of the type indicated by the descriptor). Embodiments of the invention should have a simple and extensible type system to allow for the specification of the descriptors.

[0034] FIG. **4** is an exemplary model of an annotation system that may be used to organize and annotate every blog and blog item with a descriptor. The following are some examples of how the model is used.

[0035] A descriptor attribute type **410** may be, for example, a Number, a Date, a Location, a Set, or a Classification, just to name a few, and can be further restricted by a constraint type. A constraint type **412** may be, for example, a Type, a Domain, a Size, or a Pattern (regular expression), just to name a few. In an embodiment, for example, Java implementations of interfaces defined by a blogging framework need to be provided with both the descriptor type **410** and the constraint type **412**. The exemplary system will specify proper syntax and validate a user's entries. The syntax of a constraint should be specified generally, regardless of the attribute type it is associated with. It should be understood that not all constraint types **412** can be applied to every descriptor type **410**. The relationship between descriptor type **410** and constraint type **412** is such that the descriptor type **410** can be (or must not be) constrained through the given constraint type **412**. For example:

[0036] A Number descriptor **414** can be constrained by **416**, a domain: [1 . . . 10]

[0037] A Set descriptor **414** can be constrained by **416**, a Type (for its elements) and/or Size and/or Domain:

[0038] Number—can only accept numbers

[0039] (,3)—not more than three

[0040] (a, b, c, d, e)—only these values are allowed.

[0041] A Classification can be constrained by a Domain:

[0042] (a(a1, a2, a3), b(b1, b2(b21, b22), b3, b4), c)—this is a hierarchical domain.

[0043] These instantiations will take place in the descriptor **414** and constraint **416**. When an actual descriptor is declared, a set of constraints can be associated with its attributes (i.e., a subset of the constraint type **412** associated with that particular descriptor type **410**). For example, a Rank descriptor attribute **414** can be a type Number and it imposes the Domain constraint [1 . . . 10] and can be used to annotate blog items **418** with a Rank. When a Rank descriptor **414** is defined as in this example and is instantiated for a blog item **418**, it will take a value, for example, between 1 and 10 in this case.

[0044] An unconstrained Set may allow a user to provide any group of values desired (e.g., apple, horse, p2p).

[0045] Similarly, an unconstrained Classification would allow a user to provide any group of values desired (e.g., apple/ipod/accessory).

[0046] To define a Categorization descriptor attribute **414** for a blog's items, a user may use a Classification descriptor type **410** with a constraint **416** by a Domain constraint type **412**, specified as: (work(ATT, OpenSource), sports(climbing, basketball, tennis), hobbies (cars, guitar, sociology)).

[0047] Thus, a user defines the attributes used to describe the information items provided in the user's blog. When a new information item is being added to a user's blog, the